Noisy In-Memory Recursive Computation with Memristor Crossbars

Elsa Dupraz, Lav Varshney

Noisy In-Memory Recursive Computation with Memristor Crossbars

Elsa Dupraz[†] and Lav R. Varshney[‡]
[†] IMT Atlantique, Lab-STICC, UBL
[‡] University of Illinois at Urbana-Champaign

***Abstract*—This paper considers iterative dot-product computation implemented on in-memory memristor crossbar substrates. To address the case where true memristor conductance values may differ from their target values, it introduces a theoretical framework that characterizes the effect of conductance value variations on the final computation. For simple dot-products, the final computation error can be approximated by a Gaussian distribution; the mean and variance values of the corresponding Gaussian distribution are provided. For iterative dot-product computation, recursive expressions are derived for the means and variances of the successive computation outputs. Experiments verify the accuracy of the proposed analysis on both synthetic data and on images processed with memristor-based principal component analysis.**

I. Introduction

The Von Neumann architecture in computer engineering separates memory from computation and has been a standard paradigm for decades. Yet it introduces the key communication challenge of moving data back and forth between memory and compute, a problem often known as the *memory wall*. In order to continue scaling laws of computational performance, there has been a move to break this memory wall and use in-memory computation [1], [2]. To perform computation directly within memory, several memristor-based architectures have been proposed, but the crossbar architecture has emerged as quite prominent since it naturally leads to several important computational kernels from workloads in artificial intelligence and elsewhere [3]. In addition to this new architectural approach for in-memory computing, there has also been interest in using novel low-power device technologies for implementation, such as spintronics and ferroelectrics. Unfortunately, such nanoscale devices are often noisy [2], and so there is a need to understand the informational properties of noisy in-memory computing using memristor crossbar architectures.

As noted, memristor crossbars can be used to directly implement numerous key computational kernels [4], and there are even programming languages being developed for them [5]. As an example, memristor crossbars can be used for Hamming distance computation [6], [7]. More generally, they can be used for analog dot-product computation in memory [8], [9]. Dot-product computation in memory is useful for deep neural networks [3], sparse coding [10], K-means clustering [11], and for solving optimization problems such as linear and quadratic programming [9]. Memristor crossbar-based dot-product computation is especially efficient since it is realized in only one clock cycle.

In this paper, we focus on dot-product computation in memory, which covers a wide range of applications [12], [13]. Recall that the internal conductance value of a memristor depends on the current that flows through the memristor, an electronic device property useful for computation [1]. In order to perform the computation, vector coefficients are given by input voltage values, and internal conductance values of the memristors must be set to values that depend on the coefficients of the matrix involved in the computation.

However, as described in [9], the true conductance values may differ from the target values. These variations can be seen as noise introduced in the computation, and they may affect the final computation result. The effect of these variations was theoretically studied in [7] in the case of Hamming distance computation. It was also studied in [9] for dot-product computation, but only from simulations. Therefore, the objective of this paper is to introduce a theoretical framework that allows us to characterize the effect of conductance value variations on the final computation. In this sense, we are concerned with intrinsic robustness, rather than coding-theoretic approaches to introduce robustness via redundancy [14].

In this paper, we model both the conductance values and the voltage values as independent random variables. We show that, for a large class of distributions for both the conductances and input voltages, the final computation error can be approximated by a Gaussian distribution, and we provide the mean and variance values of the corresponding Gaussian distribution.

We then consider the extended problem of successive dot-product computations, which may arise in successive layers of neural networks and which allows for fixed-point computation and for solving linear and quadratic optimization problems [9]. Here, we provide recursive expressions for the means and variances of the successive computation outputs, a little reminiscent of the extended Kalman filter and of the Gaussian approximation to density evolution [15]. These recursive expressions are obtained from second-order Taylor expansions of the means and variances. Interestingly, we show that the means are given by the exact computation outputs, and as a result, the variances provide the mean-squared errors between the noisy memristor-based computation and the exact computation result. Our experiments verify the accuracy of the proposed expressions on synthetic data and show the

This work was supported in part by the “Make our Planet Great Again” Initiative of the Thomas Jefferson Fund and by grant ANR-17-CE40-0020 of the French National Research Agency ANR (project EF-FECtive).

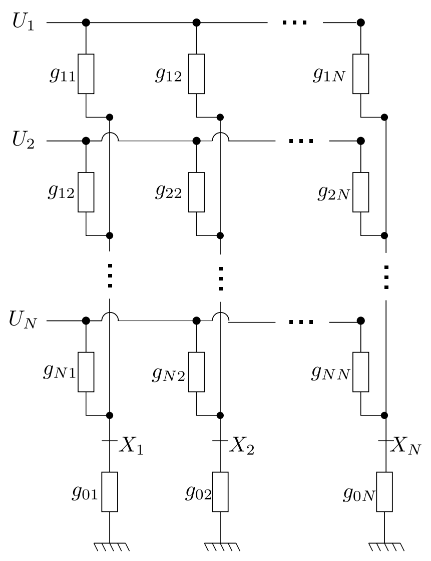

Fig. 1. Electronic model for an $N \times N$ memristor crossbar structure

influence of conductance variations on a practical application: image denoising from a memristor-based Principal Component Analysis (PCA).

Throughout the paper, uppercase letters denote random variables, and lowercase letters denote their realizations. Bold letters are used to denote vectors. Finally, $[\![1, N]\!]$ denotes the set of integers between 1 and N.

II. Computation in memory

In this section, we describe how memristor crossbars can be used for dot-product computation in memory. The electronic model for an $N \times N$ memristor crossbar structure is shown in Figure 1. In this model taken from [9], a memristor connects each horizontal Word-Line (WL) to each Vertical Bit-Line (BL). We use g_{ij} to denote the conductance of the memristor between WL_i and BL_j. We also use u_i to denote the input voltage value on WL_i, and x_j to denote the output voltage on BL_j. The value of x_j is measured through a pull-down resistor of conductance g_{0j}. We denote $\mathbf{u} = [u_1, \dots, u_N]$ the input voltage vector, and $\mathbf{x} = [x_1, \dots, x_N]$ the output voltage vector.

According to [9], output voltage x_j is related to input voltages $u_1, \dots, u_n$ by the expression

$$x_j = \sum_{i=1}^{N} \frac{g_{ij}}{\sum_{k=0}^{N} g_{kj}} u_i. \tag{1}$$

Equation (1) can also be restated in matrix form as

$$\mathbf{x} = \mathbf{u}\mathcal{G}, \tag{2}$$

where the matrix $\mathcal{G} = \{g_{ij}\}$ is of size $N \times N$. Therefore, it suffices to adjust the g_{ij} conductance values and the input voltage vector $\mathbf{u}$ in order to realize a given dot-product computation. This computation is very efficient in terms of latency as all the operations are performed in parallel.

Many machine learning algorithms can be decomposed so as to use memristor crossbars in their computations [9]–[11]. In the proposed architectures, memristor crossbars can be used iteratively by setting output values x_j as new inputs u_i for the next computation iteration. In such iterative computation, either the same memristor conductance values g_{ij} are used from iteration to iteration, or these values are rewritten in order to perform a different dot-product computation in each iteration.

However, dot-product computation from memristor crossbars suffers from several drawbacks [9]. First, all the conductance values g_{ij} must be non-negative. In order to deal with this issue, a simple solution consists of using two crossbars (one for positive coefficients and one for negative coefficients). In addition, the true conductance value g_{ij} may vary from the target value, which may introduce errors in the computation. The objective of this paper is to address this second issue. In the following, we first consider noisy dot-product computation. We then investigate the case of successive noisy dot-product computations.

III. Noisy Dot-Product computation

In order to study the influence of variations in conductance values g_{ij} on the final computation, we model these conductance values as random variables denoted as G_{ij}. In this section, in order to be able to consider a wide range of probability distributions for the G_{ij}, we only make a few assumptions on these random variables.

We assume that the random variables G_{ij} are independent, but not necessarily identically distributed, and that their first and second moments exist. Formally, this is given by the conditions $\text{E}[|G_{ij}|] < \infty$ and $\text{E}[|G_{ij}|^2] < \infty$. We further assume that G_{ij} has mean g_{ij}, and we denote $\text{E}[G_{ij}] = g_{ij}$, $\text{Var}(G_{ij}) = \sigma_{ij}^2$. In [7], [9], it is assumed that each random variable G_{ij} follows a Gaussian distribution $\mathcal{N}(g_{ij}, \sigma_{ij}^2)$ with mean g_{ij} and variance σ_{ij}^2. This model can be seen as a particular case of our analysis.

In addition, in order to take into account potential variations in previous computations, we also describe the input voltages as random variables U_i. Here again, we assume that the U_i are independent, but not necessarily identically distributed, and that the first and second moments of each U_i exist. We further denote $\text{E}[U_i] = u_i$ and $\text{Var}(U_i) = \gamma_i$. Note that in [7], [9], the input voltages u_i were considered as deterministic quantities.

The objective of this section is to propose approximations of the probability distributions of the random variables X_j that represent the outputs of the noisy dot-product computation. The X_j are obtained from (1), by replacing the deterministic quantities $g_{i,j}$ and u_i by the corresponding random variables $G_{i,j}$ and U_i.

A. Approximate distribution for X_j

Let us now prove a theorem that allows us to derive approximate distribution for a given output value X_j.

Theorem 1: Consider two sequences $(U_i)_{i\in[\![1,N]\!]}$ and $(G_{ij})_{i\in[\![0,N]\!]}$ of independent random variables and assume that all the random variables U_i, G_{ij} admit finite first and second moments. If the sequence $(U_iG_{ij})_{i\in[\![1,N]\!]}$ satisfies Lindeberg's condition (see [16]), and if

$$\alpha_j = \lim_{N\to\infty} \frac{\delta_j^2}{N^2} \neq 0, \tag{3}$$

where $\delta_j = \sum_{i=0}^N g_{ij}$, then

$$\frac{N^2}{\sqrt{v_j}}\left(X_j - x_j\right) \overset{d}{\Rightarrow} \mathcal{N}\left(0, \frac{1}{\alpha_j^2}\right) \tag{4}$$

where $\overset{d}{\Rightarrow}$ stands for the convergence in distribution, and

$$v_j = \mathrm{E}\left[\left(\sum_{i=1}^N \delta_j U_i G_{ij} - \Delta_j u_i g_{ij}\right)^2\right], \tag{5}$$

with $\Delta_j = \sum_{i=0}^N G_{ij}$.

Proof: We first express

$$X_j - x_j = \frac{\sum_{i=1}^N \left(\delta_j U_i G_{ij} - \Delta_j u_i g_{ij}\right)}{\Delta_j \delta_j} \tag{6}$$

Since $\mathrm{E}[|G_{ij}|] < \infty$ for all i, j, then, by the weak law of large numbers,

$$\frac{1}{N^2}\Delta_j\delta_j \overset{\mathbb{P}}{\Rightarrow} \alpha_j, \tag{7}$$

where $\overset{\mathbb{P}}{\Rightarrow}$ stands for the convergence in probability. In addition, since $\mathrm{E}[(U_iG_{ij})^2] < \infty$ and since the Lindeberg's condition is satisfied, we apply the central limit theorem in order to show that

$$\frac{1}{\sqrt{v_j}}\sum_{i=1}^N \left(\delta_j U_i G_{ij} - \Delta_j u_i g_{i,j}\right) \overset{d}{\Rightarrow} \mathcal{N}(0,1). \tag{8}$$

To finish, we apply Slutsky's Theorem [17, Page 19] from (7) and (8) in order to show (4).

■

First note that condition (3) in Theorem 1 may be verified in most cases since the conductance values $g_{i,j}$ are all greater than 0. It only suffices to verify that the series $(\sum g_{ij})$ is not convergent. Then, Theorem 1 permits us to conclude that the probability distribution of a computation output X_j can be approximated by a Gaussian distribution

$$\mathcal{N}\left(x_j, \frac{v_j}{\alpha_j^2 N^4}\right)$$

with parameters α_j and v_j given in the theorem. Interestingly, this shows that whatever the distribution of the random variables U_i and G_{ij}, the random variable X_j is centered around the true value x_j given in (1). In addition, the obtained values of v_j can be used to approximate the mean-squared error between X_j and x_j as

$$E[(X_j - x_j)^2] \approx \frac{v_j}{\alpha_j^2 N^4}.$$

Finally, although the convergence to a Gaussian distribution may require a large value of N, we will show in our simulations that the Gaussian distribution (8) approximates well the behavior of X_j even for medium values of N.

In closing, note that the conditions given in the theorem (finite moments, Lindeberg's condition) are satisfied by a wide range of distributions, including the case described in [7], [9] where G_{ij} is Gaussian and U_i is deterministic. In the next part, we provide the expression of v_j in (5) in the case where the inputs U_i are deterministic, and in the case where they are random variables. The provided expressions only depend on the first and second moments of the random variables U_i, G_{ij}, and therefore work for the Gaussian case considered in [7], [9].

B. Expressions of v_j

In this section, we provide the expression of v_j for the case where U_i is deterministic and equal to u_i, and for the case where U_i is random with mean u_i.

We first assume that the U_i are deterministic. In this case, v_j can be expressed as

$$\begin{aligned} v_j &= \sum_{i=1}^N u_i^2\left(\delta_j^2\sigma_i^2 + g_{ij}^2\Gamma_j - 2\delta_j g_{ij}\sigma_{ij}^2\right) \\ &+ \sum_{i=1}^N\sum_{i'\neq i} u_i u_{i'}\left(\Gamma_j g_{ij} g_{i'j} - \delta_j\left(\sigma_{ij}^2 g_{i'j} + \sigma_{i'j}^2 g_{ij}\right)\right) \end{aligned} \tag{9}$$

where $\Gamma_j = \sum_{i=0}^n \sigma_{ij}^2$. Note that, in the particular case where G_{ij} is Gaussian, the distribution of X_j can be approximated from [18, equation (9)] that provides an approximate probability distribution in analytical form for the ratio of two Gaussian random variables. In the simulation section, the approximate probability distribution of [18, equation (9)] will be compared with the approximate Gaussian distribution given in (8).

We now consider the case where both G_{ij} and U_i are random variables.In this case, v_j can be expressed as

$$\begin{aligned} v_j &= \sum_{i=1}^N \delta_j^2(\gamma_i^2 + u_i^2)(\sigma_{ij}^2 + g_{ij}^2) + u_i^2 g_{ij}^2(\Gamma_j - \delta_j^2) - 2\delta_j g_{ij} u_i^2 \sigma_{ij}^2 \\ &+ \sum_{i=1}^N\sum_{i'\neq i} u_i u_{i'}\left(\Gamma_j g_{ij} g_{i'j} - \delta_j\left(\sigma_{ij}^2 g_{i'j} + \sigma_{i'j}^2 g_{ij}\right)\right). \end{aligned} \tag{10}$$

The Gaussian approximation and the expressions of v_j given in this section hold for a one-shot dot-product computation, since they consider independent U_i. However, it is worth noting that two outputs X_j, X_j' are not statistically independent, since they are computed from the same inputs U_i. In the following, we consider iterative dot-product computation in which the outputs of one iteration serve as inputs for the next iteration.

IV. Noisy iterative dot-product computation

We are now interested in performing iterative dot-product computation by successively applying (2) with different matrices $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(T)}$ in order to compute a vector

$$\mathbf{y} = \mathcal{G}^{(T)}\mathcal{G}^{(T-1)}\cdots\mathcal{G}^{(1)}\mathbf{u}. \tag{11}$$

Such computation can be used for instance to solve linear and quadratic programming problems or to perform a PCA [9].

A particular case arises when $\mathcal{G}^{(t)} = \mathcal{G}$ for all t, which corresponds to fixed point computation. In what follows, we denote $\mathbf{x}^{(0)} = \mathbf{u}$, $\mathbf{x}^{(T)} = \mathbf{y}$, and

$$\mathbf{x}^{(t)} = \mathcal{G}^{(t)}\mathbf{x}^{(t-1)}.$$

We use $\mathbf{X}^{(t)}$ to denote the random vectors that correspond to each $\mathbf{x}^{(t)}$.

As for the simple dot-product computation considered in Section III, it is reasonable to assume that the components $X_j^{(0)}$ of the initial vector $\mathbf{X}^{(0)}$ are independent. However, after applying the first matrix $\mathcal{G}^{(1)}$, the components $X_j^{(1)}$ of $\mathbf{X}^{(1)}$ are not independent anymore. As a result, the assumptions of Theorem 1 are not satisfied when we consider the general term $\mathbf{X}^{(t)} = \mathcal{G}^{(t-1)}\mathbf{X}^{(t-1)}$. In particular, the Central Limit Theorem used in the proof of Theorem 1 does not apply to a sum of dependent random variables, unless some restrictive assumptions are made [16]. In the following, as an alternative, we provide approximated expressions for the first and second moments of the $X_j^{(t)}$, by considering the second-order Taylor expansions of these moments. Throughout this section, in order to obtain accurate Taylor expansions, we assume that condition (3) of Theorem 1 still holds.

A. Second-order Taylor expansions of the moments

In this section, we consider dependent random inputs $X_i^{(t-1)}$ ($t > 1$) such that for all $i, i' \in [\![1, N]\!]$ the covariance between $X_i^{(t-1)}$ and $X_{i'}^{(t-1)}$ exists and is denoted $\mathrm{Cov}(X_i^{(t)}, X_{i'}^{(t)}) = \gamma_{ii'}^{(t)}$. We also denote $\mu_i^{(t-1)} = \mathrm{E}[X_i^{(t-1)}]$ and $\mathrm{Var}(X_i^{(t-1)}) = \gamma_i^{(t-1)}$. We give three propositions that provide the Taylor expansions of the means, variances, and covariances of the random outputs $X_j^{(t)}$. These propositions will allow us to track the evolution of the moments of the successive random vectors $\mathbf{X}^{(t)}$.

Proposition 1: The second-order Taylor expansion of the mean $\mu_j^{(t)}$ of $X_j^{(t)}$ is given by

$$\mu_j^{(t)} = \sum_{i=1}^{N} \frac{\mu_i^{(t-1)} g_{ij}^{(t)}}{\delta_j^{(t)}} - \frac{\Theta_j}{(\delta_j^{(t)})^2} + \frac{\Gamma_j \Lambda_j}{(\delta_j^{(t)})^3} + O\left(\frac{1}{(\delta_j^{(t)})^3}\right) \quad (12)$$

where

$$\Theta_j = \sum_{i=1}^{N} \mu_i^{(t-1)} \sigma_{ij}^{2(t)} \quad (13)$$

$$\Lambda_j = \sum_{i=1}^{N} \mu_i^{(t-1)} g_{ij}^{(t)}, \quad (14)$$

and $O(\cdot)$ is the Bachmann–Landau notation.

Proof: Directly comes from the approximated expression of [19] for the mean of a ratio, derived by considering the second-order Taylor expansion of the ratio. ■

From Proposition 1, if $\mathrm{E}[X_i^{(t-1)}] = x_i^{(t-1)}$ and if condition (3) is verified, we have that $\lim_{N\to\infty} \mu_j^{(t)} = x_j^{(t)}$. By induction, this shows that when N is large enough, the random output $X_j^{(t)}$ is centered around its true value $x_j^{(t)}$, as was the case for dot-product computation considered in Section III. However, in the following, we keep the second-order terms in order to improve the accuracy of the approximation of $\mu_j^{(t)}$ when N is not too large.

Proposition 2: The second-order Taylor expansion of the variance $\gamma_j^{(t)}$ of $X_j^{(t)}$ is given by

$$\gamma_j^{(t)} = \left(\frac{\Theta_j}{\delta_j^{(t)}}\right)^2 + \frac{\Psi_j}{(\delta_j^{(t)})^2} - \frac{2\Lambda_j\Theta_j}{(\delta_j^{(t)})^3} + \frac{3\Theta_j^2\Gamma_j}{(\delta_j^{(t)})^4} - (\mu_j^{(t)})^2 + O\left(\frac{1}{(\delta_j^{(t)})^3}\right) \quad (15)$$

where

$$\Psi_j = \sum_{i=1}^{N} \left((\mu_i^{(t-1)})^2 \sigma_{ij}^{2(t)} + \gamma_i^{(t-1)} \sigma_{ij}^{2(t)}\right) + \sum_{i=1}^{N}\sum_{i'=1}^{N} g_{ij} g_{i'j} \gamma_{i,i'}^{(t-1)}. \quad (16)$$

Proof: We express $\mathrm{Var}(X_j) = \mathrm{E}[X_j^2] - \mathrm{E}[X_j]^2$ and denote $X_j^2 = V_j^2/W_j^2$. We then use the second-order Taylor expansion of a function $f(v,w) = v^2/w^2$ and apply the expectation to obtain $\mathrm{E}[X_j^2]$. Then, $\mathrm{E}[X_j]^2$ is given by Proposition 1. Note that in [19], only the first-order Taylor expansion of the variance of a ratio was provided. ■

Proposition 3: The second-order Taylor expansion of the covariance $\gamma_{jj'}^{(t)}$ of $X_j^{(t)}$, $X_{j'}^{(t)}$, with $j \neq j'$, is given by

$$\gamma_{jj'}^{(t)} = \sum_{i=1}^{N}\sum_{i'=1}^{N} \lambda_{ij}\lambda_{i'j'}\gamma_{i,i'}^{(t-1)} + O\left(\frac{1}{(\delta_j^{(t)})^3}\right) \quad (17)$$

where

$$\lambda_{ij} = \frac{g_{ij}}{\delta_j^{(t)}} - \frac{\sigma_{ij}^{2(t)}}{(\delta_j^{(t)})^2} + \frac{\Gamma_j g_{ij}}{(\delta_j^{(t)})^3} \quad (18)$$

and $\lambda_{i'j'}$ is obtained by replacing indices i,j with i',j' in (18).

Proof: We have that

$$\gamma_{jj'}^{(t)} = \sum_{i=1}^{N}\sum_{i'=1}^{N} \mathrm{E}\left[\frac{G_{ij}^{(t)}}{\Delta_j^{(t)}}\right] \mathrm{E}\left[\frac{G_{i'j'}^{(t)}}{\Delta_{j'}^{(t)}}\right] \gamma_{ii'}^{(t-1)}. \quad (19)$$

We then replace $\mathrm{E}\left[\frac{G_{ij}^{(t)}}{\Delta_j^{(t)}}\right]$ and $\mathrm{E}\left[\frac{G_{i'j'}^{(t)}}{\Delta_{j'}^{(t)}}\right]$ by their second-order Taylor expansions from [19]. ■

For finite N, we have that $\gamma_{jj'}^{(t)} \neq 0$, which shows the statistical dependency between outputs $X_j^{(t)}$ and $X_{j'}^{(t)}$.

The above three propositions allow us to track the evolution of the means, variance, and covariances of output components over iterations. It is worth noting that the expressions of the moments at time instant t only depend on the expressions of the moments at time instant $t-1$. We now present simulation results that aim to verify the accuracy of the proposed expressions.

V. Simulation results

In this section, we describe our simulation results which aim to verify the accuracy of the proposed approximations. We started with the dot product computation described in Section III. We set $N = 1000$, and generated random values of g_{ij}, U_i, σ_{ij}^2 according to uniform distributions. Then, we generated $K = 10000$ samples X_j. We plotted the histogram of these samples, and superimposed both the approximate

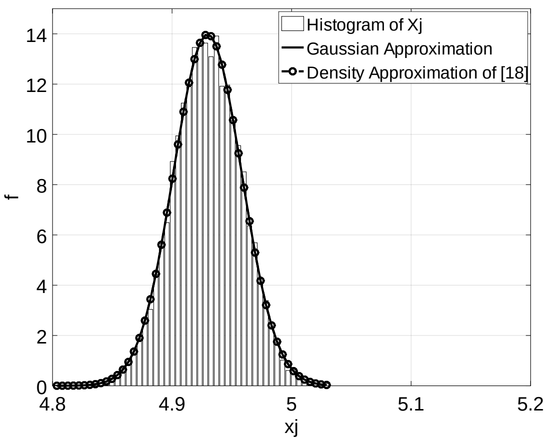

Fig. 2. Comparison of the histogram of the X_j together with the two approximated probability distributions. The Gaussian approximation and Density approximation curves are superimposed.

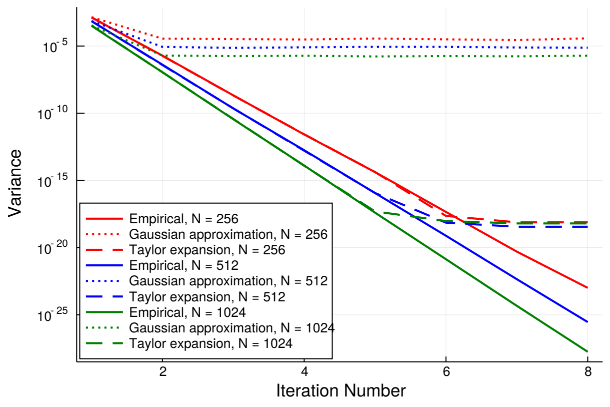

Fig. 3. Comparison of approximated variance expressions (Gaussian approximation, Taylor expansion) with empirical variance

distribution of [18] and the Gaussian distribution obtained from Theorem 1. The results are represented in Figure 2. We observe that the two approximations are superimposed and close to the histogram. This allows us to claim that the proposed Gaussian approximation accurately represents the probability distribution of the outputs (formal statistical tests are omitted for brevity).

We then considered noisy recursive computation described in Section IV. We set $T = 8$ and generated both input values U_i and 8 different matrices $\mathcal{G}^{(t)}$ at random. We considered three different values $K = 256$, $K = 512$, and $K = 1024$. In each case, we measured the empirical means and variances at each iteration. We then computed the successive theoretical approximated means and variances given in Propositions 1 and 2. For the sake of comparison, we also computed the successive Gaussian parameters obtained from Theorem 1. The results are represented in Figure 3 for the variances. We first observe that, except for the first few iterations, the Gaussian approximation does not allow us to accurately predict the empirical variance. On the contrary, the approximated variances obtained from

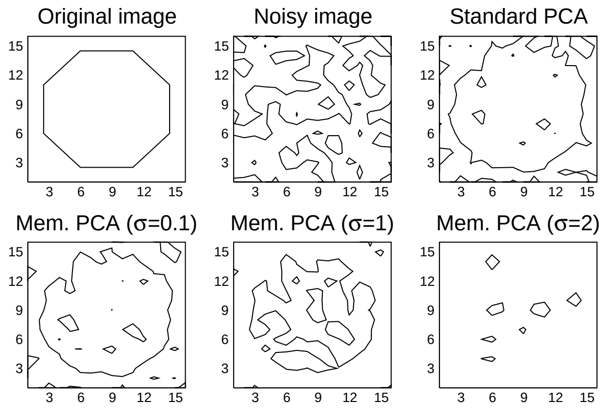

Fig. 4. Contours of original image, noisy image, and first component PCA obtained from standard PCA and from memristor-based PCA with different noise variances σ^2. We considered 10 images of size 16×16 with amplitude noise.

Proposition 2 are close to the empirical variances, for most iterations. We observe that after 5 iterations, the approximated variance saturates to a given value around 10^{-18}. This is probably due to numerical errors, because all the involved terms become very small.

Finally, we considered a practical application that is memristor-based PCA. We considered an original image of size 16×16, represented in Figure 4. We then generated 10 noisy versions of this image, where the noise is given by Gaussian random amplitudes. We implemented the memristor-based PCA proposed in [9] and applied it to the 10 images. We considered different variances $\sigma^2 = 0.01$, $\sigma^2 = 1$, and $\sigma^2 = 4$ for the memristor conductance values G_{ij}. Figure 4 shows the obtained first principal components for standard PCA and for memristor-based PCA with the different noise levels. We see that a low level of noise given by $\sigma^2 = 0.01$ produces a result close to standard PCA, while higher variance values degrade the performance.

VI. Conclusion

Understanding the properties and limits of noisy computing is becoming more and more important, now that nanoscale beyond-CMOS devices are being used in computer systems that are, themselves, no longer constructed according to the Von Neumann architecture. In this work, we have considered the key computational kernel of iterative dot-products, which arises in numerous important applications that require low energy and low latency. In some sense generalizing the work of Chen, Schoeny, and Dolecek [7], here we have developed theoretical arguments to be able to track the error in iterative dot-product computation and shown a kind of inherent robustness.

In future work, we intend to study other key computational problems in noisy memristor crossbar architectures, such as shortest path computation [20].

References

[1] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, p. 80, 2008.

[2] N. R. Shanbhag, N. Verma, Y. Kim, A. D. Patil, and L. R. Varshney, "Shannon-inspired statistical computing for the nanoscale era," *Proceedings of the IEEE*, vol. 107, no. 1, pp. 90–107, 2018.

[3] S. Jain, A. Ankit, I. Chakraborty, T. Gokmen, M. Rasch, W. Haensch, K. Roy, and A. Raghunathan, "Neural network accelerator design with resistive crossbars: Opportunities and challenges," *IBM J. Res. Dev.*, vol. 63, no. 6, p. 10, Nov./Dec. 2019.

[4] R. Gharpinde, P. L. Thangkhiew, K. Datta, and I. Sengupta, "A scalable in-memory logic synthesis approach using memristor crossbar," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 2, pp. 355–366, 2017.

[5] A. Ankit, I. E. Hajj, S. R. Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W. Hwu, J. P. Strachan, K. Roy, and D. S. Milojicic, "PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *Proc. 24th Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*, Apr. 2019, pp. 715–731.

[6] Y. Cassuto and K. Crammer, "In-memory Hamming similarity computation in resistive arrays," in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015, pp. 819–823.

[7] Z. Chen, C. Schoeny, and L. Dolecek, "Hamming distance computation in unreliable resistive memory," *IEEE Transactions on Communications*, vol. 66, no. 11, pp. 5013–5027, 2018.

[8] R. M. Roth, "Fault-tolerant dot-product engines," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2046–2057, 2018.

[9] S. Liu, Y. Wang, M. Fardad, and P. K. Varshney, "A memristor-based optimization framework for artificial intelligence applications," *IEEE Circuits and Systems Magazine*, vol. 18, no. 1, pp. 29–44, 2018.

[10] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, "Sparse coding with memristor networks," *Nature Nanotechnology*, vol. 12, no. 8, p. 784, 2017.

[11] Y. Jeong, J. Lee, J. Moon, J. H. Shin, and W. D. Lu, "K-means data clustering with memristor networks," *Nano letters*, vol. 18, no. 7, pp. 4447–4453, 2018.

[12] I. Nahlus, E. P. Kim, N. R. Shanbhag, and D. Blaauw, "Energy-efficient dot product computation using a switched analog circuit architecture," in *Proc. 2014 Int. Symp. Low Power Electronics and Design (ISLPED '14)*, Aug. 2014, pp. 315–318.

[13] N. C. Wang, S. K. Gonugondla, I. Nahlus, N. R. Shanbhag, and E. Pop, "GDOT: A graphene-based nanofunction for dot-product computation," in *Proc. 2016 IEEE Symp. VLSI Technology*, Jun. 2016.

[14] S. Dutta, V. Cadambe, and P. Grover, "Short-dot: Computing large linear transforms distributedly using coded short dot products," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 2100–2108.

[15] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Information Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[16] J.-M. Bardet, P. Doukhan, G. Lang, and N. Ragache, "Dependent lindeberg central limit theorem and some applications," *ESAIM: Probability and Statistics*, vol. 12, pp. 154–172, 2008.

[17] R. J. Serfling, *Approximation Theorems of Mathematical Statistics*. John Wiley & Sons, 2009, vol. 162.

[18] N. S. Pillai, X.-L. Meng *et al.*, "An unexpected encounter with cauchy and lévy," *The Annals of Statistics*, vol. 44, no. 5, pp. 2089–2097, 2016.

[19] H. Seltman, "Approximations for mean and variance of a ratio," *unpublished note*, 2012.

[20] Z. Ye, S. H. M. Wu, and T. Prodromakis, "Computing shortest paths in 2d and 3d memristive networks," in *Memristor Networks*. Springer, 2014, pp. 537–552.